Victorian 6502 User Group Newsletter

KAOS

For People Who Have Got Smart

OSI SYM KIM AIM ATARI APPLE UK101 ORANGE

Registered by Australia Post
Publication No. VBG4212

ADDRESS ALL CORRESPONDENCE TO 10 FORBES ST., ESSENDON, VIC. 3040

Vol.2 No.10 July 1982

Kelvin Eldridge has decided there is a need for a place for people to advertise computer/electronic items, that they want to buy or sell, so he is starting a 'Trading Post' type newsletter. See Kelvin's article on page 16. Judging by the number of items bought and sold at last month's 'TRASH AND TREASURE' sale, his venture should be very successful.

John Whitehead has offered the use of his home for local meetings, see page 13, if you want to contact John his phone number is 763 5983.

Because of lack of space, the list of Hardware available has been held over until next month. The 'Meeting was KAOS' is missing for the same reason, but the main items from the last meeting are that Tony Durrant said that the GTBUG will be available in a 4K EPROM next month and Bill Chilcott's 32K board will be ready next month. The board will cost $85.00 and the approx. cost fully populated will be $400.00. For reasons we won't go into here, there is no 'Dear Paul' column this month!

The FORTH Interest Group held their first meeting, after the general meeting, last month, but have decided it will be more convenient to hold their meeting at 1pm in future. If you are interested in FORTH or in finding out more about it, be at the school by 1pm.

The next meeting will be held on Sunday 25th at 2pm at the Essendon Primary School, corner of Raleigh St and Nicholson St, Essendon. The school children will be in as usual.

****

INDEX

PROPORTIONAL JOYSTICK INTERFACE FOR SUPERBOARD

This simple circuit, which interfaces to the Tasker Buss I/O board, was designed because of some inadequacies of the switch type joystick. It provides for up to a maximum of 6 100K potentiometers (or 3 X 2 axis joysticks). Once support software has been loaded, a simple POKE instruction is all that is required to trigger the required pot, and along with a PEEK instruction, returns with an answer in the range 0 - 255 (i.e. 8 bit answer) depending upon the position of the pot.

CIRCUIT DESCRIPTION

Once the PIA on the I/O board has been set up, a 3-bit code (i.e. 1 - 6) is output via BASIC to the LS138 (binary to decimal decoder) which via the inverters causes the trigger input of the one-shot required (i.e. 1 - 6) to go HI. A USR(X) is then performed to the pre-loaded machine code program. This program outputs binary 0 and via the LS138, inverters, sends all the one-shot inputs LO. This -ve active dge triggers the required one-shot for the duration depending upon CxR, where R depends upon the potentiometer setting. All one-shot outputs are commoned via an OR gate to PA7, (i.e. bit 7 on PIA port A input). Thus PA7 stays HI of course, for the duration depending upon CxR. Obviously only one one-shot (pot) can be triggered at any one time. The software then continually checks bit 7, and if still HI, adds one to a software counter (max count = 255 (FF)). If bit 7 goes LO, then the answer is stored and returns to the BASIC program.

SOFTWARE

The machine language program listed below can be located anywhere you like. I use a free area used for disk operations starting at 0222 hex (546 dec) because it is unaffected by BASIC. My answer is stored in 0240 hex (576 dec). Location C004 of course, refers to the PIA output register A address.

```
0222      LDA  $FF      )      Counter offset
          TAY           )

          LDA  #$00     )
          CLC           )      1 shot trigger (-ve edge)
          STA  $C004    )

LOOP      LDA  $C004    )      Is one-shot HI? (by = 1)
          ROL           )
          BCC  OUT             No, the COUNT finished, jump out
          INY                  Yes, then increment count.
          CPY  #$FF            Is count = 255?
          BNE  LOOP            No, then try again

OUT       TYA
          STA  $0240           Store COUNT in 576 dec.
          RTS                  Return to BASIC
```

CONSTRUCTION

Since the circuit is relatively straight forward with only 7 IC's, construction is no problem using VERO board, and use a 24 pin header to interface to the I/O board PIA socket. But please, take note of the NOTE on the circuit diagram.

Next page please

****

SIMPLE SOUND EFFECTS

Sound can certainly improve an arcade game and provide tactile feedback. A simple way to get it is to connect a wire to the Superboard "Noise Port" and wrap the other end around the serial of a FM Radio tuned to 100MHz. Fine tuning of the radio will provide a rich array of effects to choose to suit the game. The advantage of tactile feedback is that the player knows when a key has been pressed and this lessens the tendency to push the keys through the board in moments of excitement.

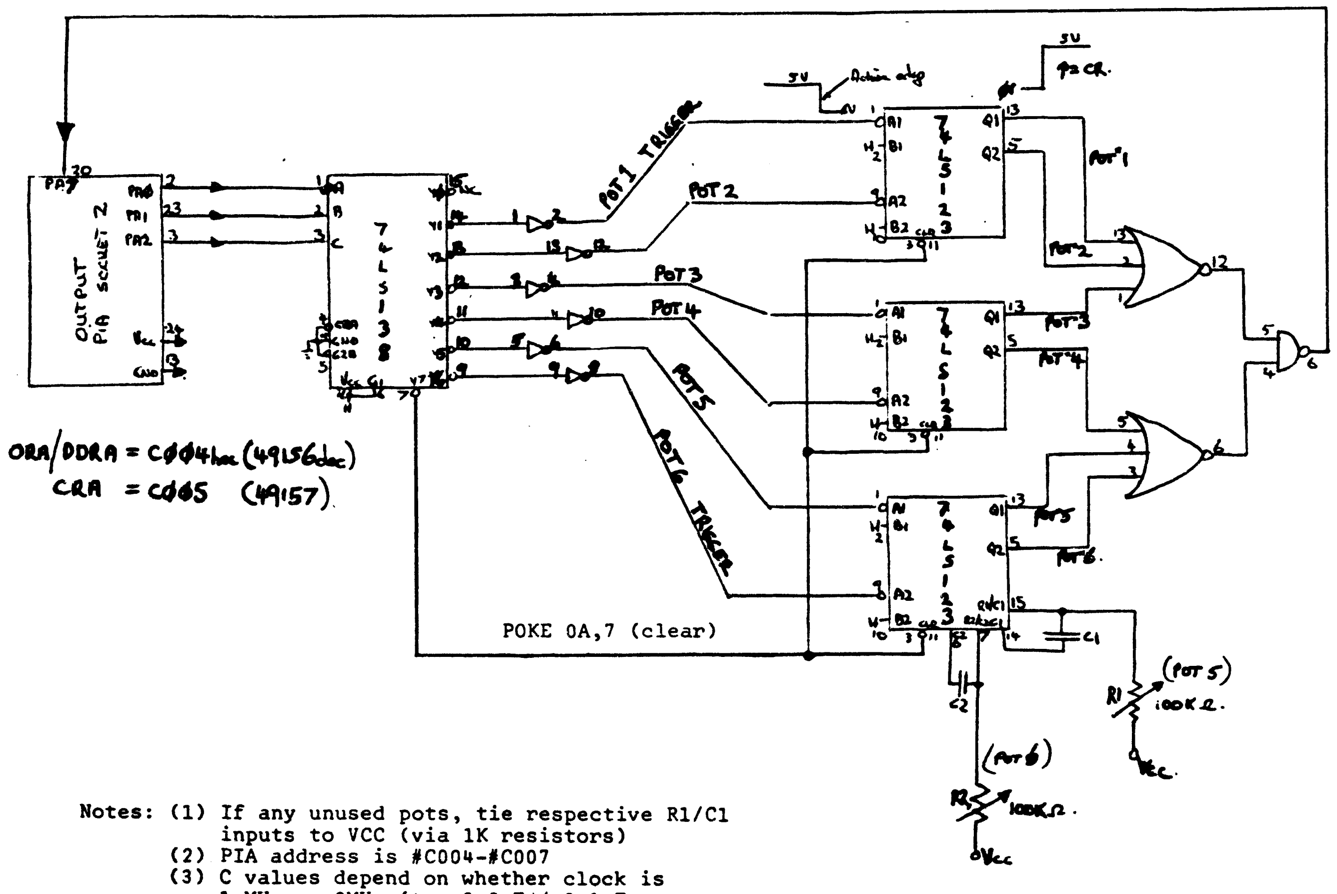
John Olding

Below is a demonstration BASIC program which includes comments in order to assist understanding.

```
 0    REM*** PIA SET-UP ***
 1    OA=49156:CA=49157
 2    REM OA=ORA/DDRA,CA=CRA ON PIA
 3    POKECA,0:REM SELECT DDRA VIA CRA
 4    POKEOA,127:REM SELECT b7 AS INPUT ON PIA
 5    POKECA,4:REM SELECT ORA on PIA VIA CRA
 6    REM** LOAD MACHINE LANGUAGE PROGRAM **
10    DIMD(25):AD=545;FORQ=1TO25:READD(Q):POKEAD+Q,D(Q):NEXTQ
20    DATA169,255,168,169,0,24,141,4,192,173,4,192
30    DATA42,144,5,200,192,255,208,245,152,141,64,2,96
40    REM*** SET up X=USR(X) START (0222) ***
41    POKE11,34:POKE12,2
49    REM
50    REM*** ACTUAL PROGRAM ***
51    INPUT"POT NO.1-6";R
52    POKEOA,R:X=USR(X):PRINT PEEK(576)
53    GOTO51:REM GOTO 52!!
60    REM * TO CLEAR 1-SHOTS-POKE OA,7
```

__NOTE__ Lines 0 - 51 need only be done __ONCE__ during set-up of program.

Paul Coburn



Notes: (1) If any unused pots, tie respective R1/C1 inputs to VCC (via 1K resistors)
(2) PIA address is #C004-#C007
(3) C values depend on whether clock is 1 MHz or 2MHz (try 0.2uF / 0.1uF

****

After purchasing the LOOKY VIDEO R.F. modulator, I found that I was unable to obtain a stable picture on my T.V. set. On turning up the sound on the T.V. set I found there was an annoying hum being produced. This made me suspect the lack of filtering in the plug pack supplying the modulator. I placed the closest electrolytic capacitor at hand (1000uF 25V) across the plug pack output and a perfect, stable picture resulted.
This info may be of some use to KAOS members.

Vic Kaprilian

Superboard

Newsletter of the Ohio Superboard User Group, 146 York St., Nundah, QLD. 4012.

I welcome all OSUG members to our pages in the KAOS Newsletter. I have been here for a couple of months now, with items from past newsletters, and have been pleased to see some recent articles for cassette based systems from KAOS members,

One of the reasons for the pleasure is that OSUG has an easy to install and use, replacement chip for the BASIC 4 ROM with the following features:-

(1) Save and Load named programs, in BASIC or Machine Code.
(2) Even at 300 Baud, 33% faster for Basic, 300% faster for M/C.
(3) With simple mods, 600 & 1200 Baud even with a 1MHz CPU clock.
(4) Will work at 4800 Baud if you do the fancy hardware.
(5) The ability to Load and Save programs in the old format is retained.
(6) Works with any known Monitor except WEMON.
(7) Works with disk systems using ROM BASIC incl. Picodos and Hexdos.
(8) A special feature enables recovery of your program after it hangs up.
(9) Two compatible versions are available to suit ACIA at $FØØØ and $FCØØ.
(A) The Cl version will AUTO-RUN programs, selectable when you Load.
(B) The Cl version will Verify that a program has saved OK.
(C) The Cl version will read program filenames without loading.
(D) The price of either version is $12.80 incl. pack & postage.
(E) Profits go to the User Group and are thus returned to members.

OSUG also has a replacement for the BASIC 3 ROM with a complete string bug fix. This is superior to other BASIC 3 replacements available, and costs $9.80.

ADDRESSING AND DECODING

The 6502 has 16 address lines, AØ to A15. The following table shows their significance.

AØ	A1	A2	A3	A4	A5	A6	A7	A8	A9	A1Ø	A11	A12	A13	A14	A15
1	2	4	8	16	32	64	128	256	512	1024	2048	4096	8192	16384	32768

Now this is a Binary progression. If you wanted to decode all bytes in a 2k memory, you would need 11 address lines, AØ to A1Ø. Add up all the numbers from A1Ø down and you will get 2047, plus, of course byte Ø. Thats 2k.

Internally, the memory chip will decode these address lines so that any byte within the chip can be selected with a unique signal on the 11 address lines.

Some means of selecting just which chip we require is needed because in a system like the Superboard, there will be many chips sharing those first ten or eleven address lines. AØ to A9 goes to each of the 16 or more RAM chips, and AØ to A1Ø goes to each of the 5 or more ROM or EPROM chips in the system.

This is where external decoding comes in, to select which one of these devices we wish to read from or write to. The Superboard uses several special decoding devices, as well as common logic chips, to service this need.

In Newsletter 24, I included a diagram of a useful decoding chip, the 74LS138. Depending on the way it is configured, the LS138 can decode various blocks of memory. The most useful ones being 1k, 2k, 4k, and 8k, because there are readily available devices (at reasonable prices) with this much memory.

OSI also use the LS139 and LS157 for decoding. L = low power. S = high speed.

Superboard

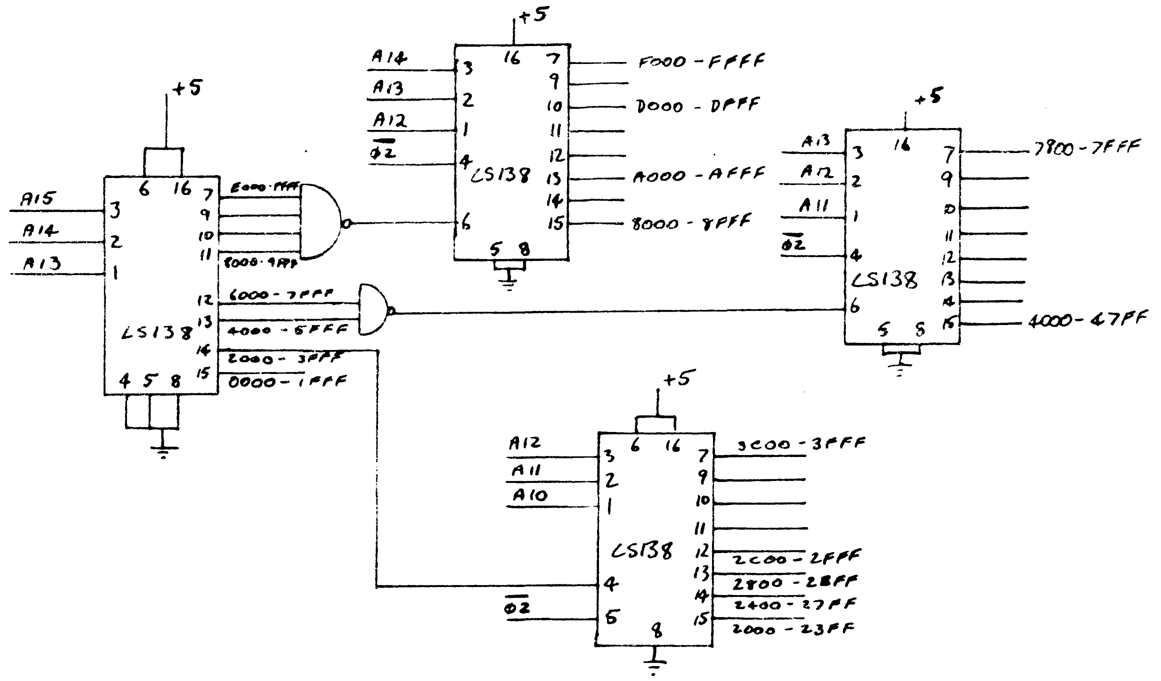
Using the Newsletter 24 diagram, you will be able to trace out all the signals in the LS138, and see just what we are doing. Prepare your own truth table.

Here is a simple example:-

	AØ	A1	Address
Both Lo	0	0	0000
	1	0	0001
	0	1	0002
Both Hi	1	1	0003

As you can see, AØ & A1 can decode 4 addresses including byte zero.

An interesting possibility now that 2732 EPROMS are around $7, would be to have two monitors in the one chip, and with the same address. By holding pin 21 (A11 on a 2732) low, you would select the lower 2k, and by switching it high, you would use the upper 2k. Could be useful if you're contemplating the TASAN 32 x 64 board and don't want an annoying little routine to have to load each time you want the 24 x 24 format that most of the programs are written in.



8k decoder   4k decoder   1k decoder   2k decoder

In the diagrams above, LS138s are decoding for several situations. Note the inverted logic used in the nand gates. Here the logic used is an "or" function. If any input to the gate is Ø, the output is 1 and selects the appropriate decoder. I haven't shown pin 15 of the 8k decoder used. It could easily go to the $\overline{\text{CS}}$ pin of an 8k RAM or ROM. Both of these devices are in production, and the 40 pin 8k EPROM is under $20.

The devices above are fully decoded, but there is not always the need to fully decode. In Bernie Wills' IPROM article last month, you could use the output pins of the LS138 to select addresses 8k-10k, 16k-18k, 24k-26k, 32k-34k, 40k-42k, 48k-50k, and 56k-58k, but nothing inbetween. This is a perfectly satisfactory arrangement because the first four addresses would most likely be where your RAM would end. However the higher addresses would clash with the BASIC ROM addresses and the disk area.

Ed Richardson

HIGH SPEED TRIG FUNCTIONS

A major problem with 3-D displays on graphics VDUs is the need for sine and cosine functions. If the display is moving they need to be calculated rapidly. Many people use BASIC and just put up with the lack of speed. This article presents a method of generating sine and cosine functions quickly and easily and can be applied to Machine Code.

Before discussing the technique, we should first find the requirements for a moving 3-D display. As mentioned earlier, speed is probably the most important factor. When projecting a 3-D image on a 2-D VDU screen we find that both the sine and cosine of a given angle are requirec The algorithm given in this article generates the sine and cosine simultaneously. Accuracy is important, but the 9 digit accuracy given by BASIC is a little ridiculous. An interesting feature of 3-D displays is that the angles change by a small amount as the object moves. This is the feature that will allow the algorithm to work.

The algorithm has its origins in differential calculus. Those familiar with calculus and Machine Code will know that normally, the two do not mix. The principle involved can be extended for generating exponentials and many other complex functions.

Differential calculus deals with the rate of change of a function. Knowing the rate of change of a function at a particular point, we can predict the value of that function a little further on with a fair degree of accuracy.

The derivative of a function is another function which gives the rate of change at any point. The derivative of Sin(x) just happens to be Cos(x) and the derivative of Cos(x) is -Sin(x). We know that when Sin(x) equals 0, Cos(x) equals +1 or -1. If we initialize a variable Sin to zero and Cos to one (ie. the angle x=0), the current value of Cos can be used to approximate the next value of Sin. The new value of Sin allows us to approximate the next value of Cos. In BASIC, this can be done in the following way:-

```
 10  SIN=0:COS=1
 20  FRAC=256
     ------
     ------

500  SIN=SIN+COS/FRAC
510  COS=COS-SIN/FRAC
     ------
     ------
```

The variable FRAC controls the rate of increase of the angle. It also affects the accuracy of the algorithm. The smaller the value of FRAC, the faster the function oscillates and the less accurate the approximation becomes. If FRAC is too high the function may stop by causing SIN/FRAC or COS/FRAC to equal zero.

Applying this algorithm to Machine Code still presents a problem. Machine Code can not handle division very easily, but if FRAC is selected carefully, the problem can be overcome. In Machine Code, division by two is just an LSR instruction. Division by four is two LSR instructions. If SIN and COS are set up as 16 bit variables, division by 256 is simply a matter of taking the high byte. Remember that the sine and cosine functions extend to plus and minus one, so the arithmatic must be signed two's comlement.

Since the errors due to the approximation accumulate as more and more values are taken, some 'stablizing' must be built into the program to reduce them as much as possible. For example, when either variable crosses zero, the other one could be set to +1 or -1 depending on where it is in the cycle. It would also be a good idea to set the zero crossing variable.

Another problem with the algorithm is that it will allow SIN and COS to exceed +1 or -1. Therefore, the assumed decimal point should allow for the possibility of overflow. This means that the most significent bit will be the sign bit, the next bit is needed for overflow. This means that $2000 will be equivalent to +1, and $E000 will be equivalent to -1.

If you are having difficulty accepting a technique this simple creating a function as complex as a sinusoid, then try the following BASIC program. This program will generate a series of asterisks in a sinusoidal pattern.

```
 10  LETS=0
 20  LETC=1
 30  LETF=256
 40  LETA=0
 50  LETK=50
500  LETS=S+C/F
510  LETC=C-S/F
515  IFA/K<>INT(A/K)THEN525
520  PRINTTAB(32+20*S);"*"
525  LETA=A+1
530  GOTO500
```

Using a similar principle, BASIC's EXP(X) can be approximated given that the derivative of EXP(X) is EXP(X) and EXP(0) = 1.

Those who know a bit of calculus will be able to find many more functions which can be approximated in a similar way.

Brian Campbell

****

A COUPLE OF QUICK TIPS FOR DISK USERS:-

a) By using the Editor in the Assembler you can renumber a BASIC program ie. call in the Assembler, then load your BASIC program and do an 'R', this will renumber your lines.

b) The Editor in the Assembler can also be used for a block delete as you can call the program into the Assembler Editor and delete a large block quicker than you can do it in BASIC.

Nigel Bisset

****

Here is a program I have written to remove blocks of lines or to remove old programs without removing Utility programs that are in memory at the same time.

```
60000  INPUT'REMOVE LINES FROM';X:INPUT'THROUGH TO';Y:B=256:A=769
60010  PRINT:PRINT'       REMOVE FROM'X'TO'Y:PRINT
60020  IFY > 59999THENSTOP
60030  INPUT'ARE YOU SURE';A$:IFLEFT$(A$,1)< > 'Y'THENRUN
60040  T=PEEK(A+3)*B+PEEK(A+2):IFT=XTHENZ=A
60050  IFT=>YTHEN60070
60060  A=PEEK(A+1)*B+PEEK(A):GOTO60040
60070  P=PEEK(A):K=PEEK(A+1)
60080  IFT>YORZ<769THENPRINT'LINE NOT FOUND':STOP
60090  POKEZ,P:POKEZ+1,K
60100  PRINT'TYPE  'X'  AND PRESS RETURN AFTER OK':END
```

John Whitehead

FORTH INTEREST GROUP

After the last General Meeting, a meeting of those interested in the FORTH language was called. Twenty two people put their names down as "interested". A general discussion took place on what FORTH was and why use it. I summarize ---

WHAT IS FORTH?

FORTH is a software package which, in 5K to 10K bytes of code -

* is conversational like APL, Lisp, or Basic;
* includes a compile facility with many high-order language, structured-programming constructs;
* exhibits performance very close to that of machine-coded program equivalents;
* is written largely in itself and consequently is largely portable;
* places no barriers among combinations of system, compiler, or application code;
* can include an intergrated, user-controlled virtual memory system for source text and data files;
* permits easy user definition of new data types and structures; and
* can be extended to include new commands written either in terms of existing commands or in the underlying machine language (an assembler for the underlying machine can be implemented in as little as a page of text of existing commands).

WHY CHANGE FROM BASIC? One member wrote; "I tried it because I got fed up with BASIC's speed and limitations and I like a change from Assembler".

WHAT EQUIPMENT DO I NEED? Minimum- 6502 computer (FORTH is available for many CPU's) 16K RAM cassette. FORTH in ROM is available for OSI computers.
BETTER- 24K and floppy disk(s)
FORTH is available from KAOS for all OSI computers.

WHAT DOCUMENTATION IS AVAILABLE?

1. fig-FORTH INSTALLATION MANUAL - contains a glossary of FORTH Words (commands) and a listing of the entire FORTH language written in FORTH. Indispensable to serious users.
2. OSI-FORTH User Manual - describes the Technical Products implementation of FORTH for OSI; also contains hints on use of Editor and Assembler. Contains listings of FORTH system sections required to run FORTH on OSI computers and utilities.
3. FORTH articles in Byte magazine. The August 1980 edition of BYTE had a special feature on FORTH. Good basic descriptions of FORTH.
4. FORTH Assembly Listing. An Assembly listing of the fig-FORTH system. Good reference - good for the diehard fanatic who wants to key in 80 pages of Assembler or 6K of hex code.
5. FORTH TOOLS User Manual. A fig-FORTH implementation but with own video, keyboard and disk drivers. Throw away your 65D3.1.
6. Assembler listings for FORTH TOOLS I/O drivers, includes a 'real' keyboard routine that properly decodes all the keys.
7. Tiny Pascal in fig-FORTH - compiles a subset of Pascal into FORTH, listing only.
8. METAFORTH - a compiler compiler, cross-compiler. But for 8080 systems. Listing only.

WHAT IS THE BEST BOOK AVAILABLE ON FORTH?

In my opinion, they rank:-

1. STARTING FORTH by Leo Brodie, Prentice-Hall 1981. $21.50 from Mcgills, Technical Bookshop, etc.
   Starts from the very first principals, very well explained and presented, clever cartoons and works up to some 'real' programs.
2. THREADED INTERACTIVE LANGUAGES by R. Loeliger Byte Books $25.00
   Detailed inner workings. Examples in Z80 code.
3. INVITATION TO FORTH Petrocelli Books, New York 1981
4. DISCOVER FORTH Thom Hogar, Osbone McGrowhill, $17.50 from Dick Smiths.

Most of these books are available from the larger book shops. If you have trouble obtaining any of them, contact KAOS and they should be able to tell you where they are available. (FORTH is becoming very popular, which means some books are sometimes in short supply.)

IS FORTH EASY TO LEARN?
FORTH is not as easy to drive as BASIC. You need to read the book FIRST, understand how it works, before you can use it well. It is worth the effort.

Does FORTH have an active interest group which puts out a very high standard newsletter with lots of interesting articles and developements in FORTH? Yes. "FORTH DIMENSIONS" available from KAOS.

Next meeting of the FORTH Interest Group, 1pm on meeting day, will include a FORTH demonstration on the clubs new "cinema-scope" widescreen monitor.

David Wilson

****

FIX FOR EM/DOS INTERFERENCE

A small patch can be made to the DOS, so that using the Extended Monitor will not interfere with stepping the disk drive head. The problem is that the EM and the DOS disagree as to the purpose of location $EF. This dispute is won by the EM, which stores something there (I don't know what), which is later used by the DOS as its step-rate.

The fix, rather obviously, is to restore STEPRT to the value required by the DOS (08). I did this with a patch placed over the BASIC device 8 handler, whatever that is.

```
$2A86   20  B1  24  JSR  24B1    ; call patch

$24B0   60          RTS          ; so BASIC won't crash
 24B1   8E  E5  2C  STX  BUFOFS  ; do what $2A86 would have done
 24B4   A9  08      LDA  #8
 24B6   85  EF      STA  STEPRT  ; restore step rate
 24B8   60          RTS
```

This was implemented on an OS-65D V3.2 with an 8" disk drive.

Peter McLennan

****

FAST REPEAT FOR KEYS

Editing using DABUG can be speeded up dramatically using the following MC program. The <CTRL> A is made to repeat at a high rate, tracing across a line of BASIC at twice normal speed. The program works by interfering with the keyboard routine's repeat rate flag at $0214. Speed is selectable by POKEing a value between 20 and 1 into 559.

After warm starting, first wake up DABUG, then POKE 536,34:POKE 537,2. Any key or key sequence can be made to repeat quickly if its ASCII code is put into $0227=551.

```
0222 48                PHA
0223 AD1302            LDA $0213    ;save A acc.
0226 C901              CMP #$01     ;is character <CTRL> A?
0228 F004              BEQ REPEAT   ;Yes-- fast repeat!
022A 68       GO       PLA          ;get A acc. back
022B 4C00F8            JMP $F800    ;go to input routine in DABUG
022E A90A     REPEAT   LDA #$0A     ;get repeat speed; 1 is fastest
0230 8D1402            STA $0214    ;store repeat rate
0233 4C2A02            JMP GO       ;go to DABUG
```

Stuart Thomas

SYSTEM RAM ADDRESSING

As discussed, the interrupt vectors are located at the top of the system RAM. Anyone familiar with the 6502 knows that the vectors are located at the top of memory, so why is the SYM different?

The fact of the matter is that the vectors have not been moved but echoed. When addressing memory normally, all address take part in selecting each device. The low address lines are connected to the address lines on the memory device and the remaining lines are combined to form the 'Chip select' signal. However, if an address line is not included in either the addressing or selection, the device will appear in two places; the first where the line is zero and the other where it is a one. The more address lines that are ignored, the more 'aliases' that device will have. This is why three 6522s and a 6532 take up 4K between $A000 and $AFFF.

Most of the time, aliases are created purely to cut costs by reducing the amount of address decoding circuitry. A special addressing circuit is used to generate aliases of the system RAM in the range $F800 to $FFFF and there are no prizes for guessing which system RAM locations fall in the 6502 interrupt vector area.

If this alias was removed, another 2K of addressing space would be available for more RAM or ROM. This is yet another typical SYM modification: cut a strap. The strap to be cut is the small one labelled U - 22 located about an inch below the Synertek logo. This does present a slight problem. There is now no IRQ or NMI vector. There are two possible solutions.

The most obvious solution is to replace the new 'hole' with more RAM and reserve the top addresses for the vectors. Unfortunately, this would make the SYM incompatible with the existing software which would use system RAM for the interrupt vectors. The first problem you would encounter is that the debugger would not work.

The other solution is to fill the 'hole' with a ROM and point the IRQ vector to a JMP ($A67E) instruction, and point the NMI vector to a JMP ($A67A) instruction. This would slow the response to an IRQ or NMI by 5 microseconds, but otherwise there will be no change.

SUPERMON - EXTENDING SUPERMON

As described in the April issue of the newsletter, SUPERMON uses the vector URCVEC for calling the error printing routine when an unrecognised command is entered. If this vector is altered, you may add your own command processing routines and therefore add your own commands.

When the jump is made, the command is stored in the system RAM in LSTCOM, PARNR, P1, P2, and P3. The accumulator contains the command name (ie. the contents of LSTCOM). Those with SUPERMON V1.0 must add -

```
CMP  LSTCOM
BNE  NONHEX
```

at the beginning of the command processing routine to separate an unrecognised command from the entry of a non-hex character. Those with V1.1 will have this done for you already. In this case, the jump is made via URSVEC instead.

Those with V1.1 may wish to control the processing associated with the entry of a non-hex character as a parameter. This can be done using URSVEC in a similar way. A possible use is to get the character '%' to indicate a binary number, ie.

```
M %10010,200,2FF
```

could accept the first parameter as a binary number and the other two as hex.

Next page please

If your routine must report an error, set the accumulator to the error number and

```
SEC
RTS
```

If the command is successful, return to SUPERMON with

```
CLC
RTS
```

There are a few restrictions governing the choice of the command name. Obviously the command name and the number of parameters required must not previously exist in SUPERMON. For example, the command 'D' followed by one parameter can not be used. This is the SUPERMON 'Deposit' command. The command 'D' followed by two parameters can be used since SUPERMON regards this as an error. Control characters may be used, with the exception of NULL, Carriage Return and RUBOUT.

As you can see, the technique for extending SUPERMON is simple but an understanding of how it works is essential.

Next Month - THE DEBUGGER

The DEBUGGER is an extremely simple and grossly underrated piece of hardware. Those that have used it know how handy it is for debugging machine code subroutines. Combined with a piece of clever software, the debugger has the potential to be one of the most powerful tools of its type.

Brian Campbell

****

MODIFICATION FOR "COMP-DOS 1.2"

After a very frustrating start (the first DOS I bought didn't work, sorry George) and a few late nights, I came up with a very simple answer to Garnett Znidaric's (newsletter Vol.2 No.7, page 3) problem.

After you have initialized your system as explained in the manual, do a few modifications to BEXEC*.

For screen size 48 X 12

delete line No's 6,7,680,690
add new line :
```
535 DISK!"DR"
```

For screen size 24 X 24

as above with an extra line
```
695 RETURN
```

<u>DON'T FORGET</u>

```
DISK!"DE BEXEC*"
DISK!"PU BEXEC*"
```

After you have finished that BREAK, reload and get a pleasant surprise.

Rob Bretterecker
VK3VFP

****

CORRECTION:

"MORE ON DATES" KAOS N/L Vol.2 No.9 P.16 Frank Nicholls

```
1010  DD$="":MM$="":YY$"":DX=0:REM NULLS
1060  ONDX+1GOTO1070,1080,1090
1300  A=24569:DT$=RIGHT$(STR$(PEEK(A)+100),2)+"/"
1310  DT$=DT$+RIGHT$(STR$(PEEK(A+1)+100),2)+"/"
1320  DT$=DT$+RIGHT$(STR$(PEEK(A+2)+100),2)
```

MACHINE LANGUAGE PROGRAMMING
David Dodds

Recently KAOS has been running articles on machine language programming. Unfortunately a lot of those interested do not seem to have the fundamental knowledge required. For these people we will be presenting a series aimed at the beginning machine language programmer. During the series BASIC will be used to help explain the various concepts involved.

A knowledge of logic (AND,OR etc) and of the binary number system will be assumed (your favourite BASIC text book shoul tell you all you need to know).

A computer instruction is a pattern of binary bits (a binary number if you prefer). A machine language program consists of a series of these instructions and binary data. This is also known as an object program.

Working with binary or object code is not an easy matter. Some typical problems include:-

-programs are difficult to understand and debug
-programs are slow to enter into the computer as each digit has to be individually set
-programs are not easily readable by humans.

As an illustration examine the columns of code below. There is a difference in one bit of the code on the right hand side. Try to find it!

```
10100101          10100101
01100000          01100000
01100101          01110101
01100001          01100001
10000101          10000101
01100010          01100010
```

In binary a four digit number may have any value between 0 and 15. By using a number system covering the same range, one digit can be used to directly represent four binary digits. This number system is known as hexadecimal.

Hex numbers use a mixture of alpha and numeric characters. The characters 0 to 9 have the same values as they do in decimal while the characters A,B,C,D,E and F represent the values from 10 to 15 respectively.

Our two columns of binary code, when expressed in hexadecimal become:

```
A5     A5
60     60
65     75
61     61
85     85
62     62
```

The mistake is far more obvious!

Hex numbers are prefixed with the $ symbol to identify them as hexadecimal values.

With four hex digits we can specify any location in memory that the computer can access. Two hex digits can directly represent an instruction.

While hex may at first seem a little strange, it is easy to work with and does provide a convenient method for loading data into the computer. To simplify working with hex special programs built into the monitor in OSI machines carry out the task of translating the hex characters into the binary patterns that the computer uses.

Microcomputers have four basic building blocks or features:

1.	Input device(s)	-Tape or keyboard
2.	Memory	-RAM and ROM
3.	Microprocessor	-the 6502 in our case
4.	Output device(s)	-Tape, screen etc

The microprocessor has the ability to handle calculations, make decisions and also access (read) and modify (write to) memory. This facility gives the microprocessor the ability to alter its own instructions.

Fig. 1 shows the interrelations between the various parts of a microcomputer.

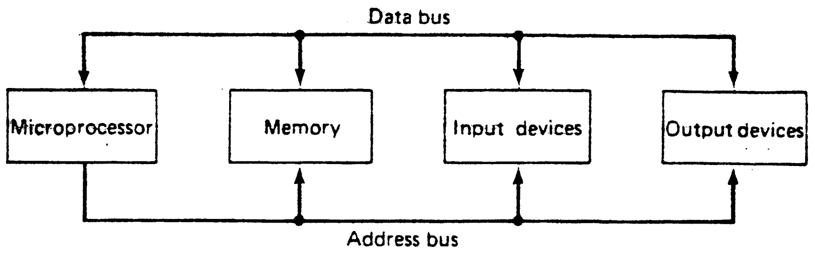


Figure 1 *Block diagram of a microcomputer system sharing data and address buses*

In operation the microprocessor can specify any memory location (or I/O device) by placing a value on the address bus. The value to be stored is then placed on the data bus by the microprocessor. During a read operation the input device or memory places its contents on the data bus where it becomes available to the microprocessor.

The main feature of a microprocessor is the central processing unit or C.P.U. This section interprets and carries out the instructions passed to it from memory along the data bus. The CPU contains

(a) an arithmetic/logic unit (ALU) for arithmetic and logic operations
(b) a control section
(c) various data and control registers for temporary storage and manipulation of data and instructions

In the 6502 most of the registers are 8 bit (8 binary digit). We do not need to know all of the registers in the CPU and will deal with only six. For our purposes we will consider each of the registers as if it were a BASIC variable.

(1) A = ACCUMULATOR: This high speed messenger and work horse is both input and output to the ALU.
(2 & 3) X & Y = INDEX REGISTERS: used as counters, pointers and temporary storage
(4) P = (PROCESSOR) STATUS REGISTER: this is actually a collection of independant one bit registers. Each bit could be likened to an element in an array variable. This register will be dealt with in more detail in a later article.
(5) S = STACK POINTER: A special area of memory called the stack is used for 'housekeeping' during the running of a program. This register points to the most recent entry on the stack.
(6) PC = PROGRAM COUNTER: during the running of a program this 16 bit register indicates the next memory location to be accessed. The PC steps sequentially through memory one byte at a time unless instructed to do otherwise

The processing of an instruction is carried out in two phases:
(a) fetch the instruction
(b)interpret and execute

Next month we will start to look at the types of instructions available to us on the 6502.

****

In response to last months call for small gatherings, any KAOS member can call in for a chat or exchange tapes at my place (no smoking) on the Friday two weeks (12 days) after a KAOS meeting, at 7pm.
I have an 8K Superboard with 8k CMOS RAM and 8K EPROM on a Tasker Buss.

John Whitehead

I have recieved a lot of enquires from members who have fitted the TASAN video board and now would like to make their keyboard C1-C4 switchable. Below is a ciruit which will allow this to be done with minimal changes to the superboard. The only changes to be made are fit sockets to U2, U3, U4, U5 if they are not already socketed.

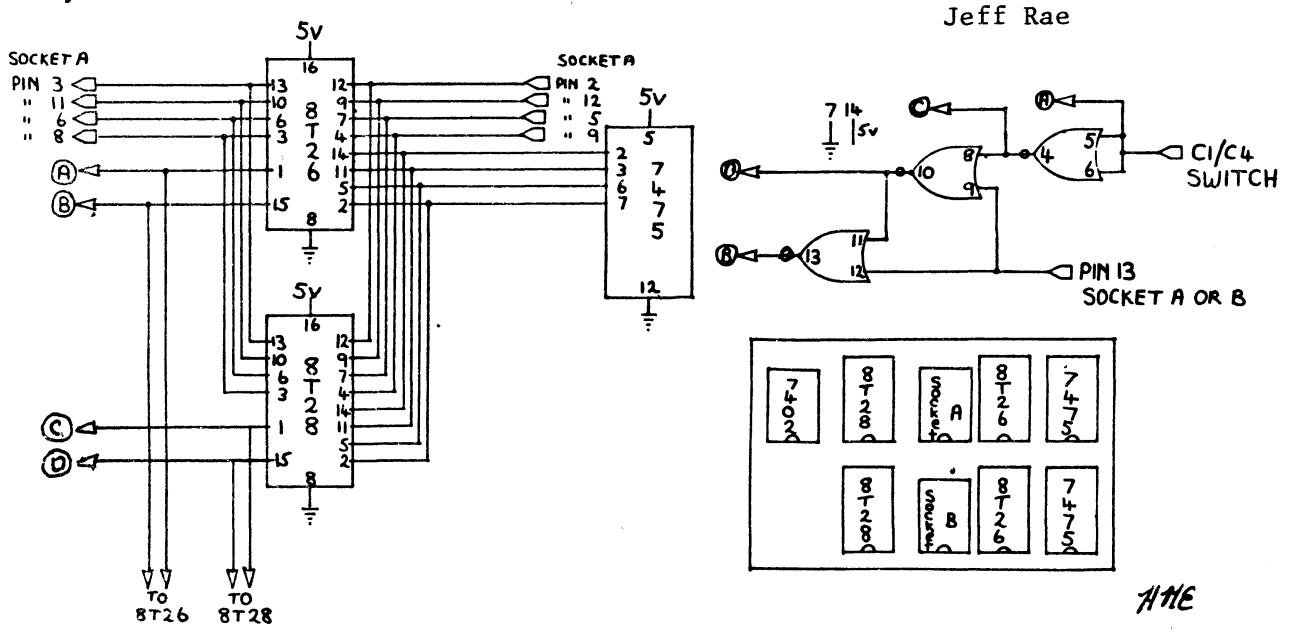
The circuit is constucted on a piece of 0.1 spacing matrix board using wirewrap sockets. The sockets are fitted to the board as shown in the drawing, making sure that the pins line up with the sockets on the superboard and wired as shown in the circuit. Note that only one half of the circuit is shown, both sides are wired exactly the same. After you have finished wiring the board cut pins 2, 3, 6, 7 of the 7475 sockets so that they don't plug into the sockets on the superboard. Make sure that all other pins clear components on the board.

The point marked C1-C4 is connected to the same switch as the video board. To make this modification usable you are advised fit a 4K monitor ROM with a C1 and a C4 DABUG as per the article in Kaos Vol.2 No.7 page 14.

The TASAN video board also needs a mod to enable it to be switched to the C1 format while it is in either of the C4 formats.
First cut the track from pin 3 of U51 .
Connect a wire between pin 5 and pin 8 of U29.
Connect a wire between pin 3 of U51 to pin 6 of U29.
Connect a wire between pin 7 of U29 to pin 1 of U5.

Now when you switch to C1 there will be no need to do the POKE as well if you are in 64 screen format.

Jeff Rae



For members using my serial to parallel printer driver it is possible to speed up the transfer rate because it has a handshaking line to tell the computer when to stop. The easiest way to impliment this is type the following line direct or as part of a program.
POKE61440,3:POKE61440,0
To disable this feature so that you can use cassette type.
POKE61440,3:POKE61440,17

Another handy hint for printer users, location 15 (DEC) is used to set the number of characters across the page.
eg.If your microline is set up for a 64 character line and you send 72 you will find that it overwrites the same line. The fix is to change the length of the line sent by POKE15,62. Restore with POKE15,255.

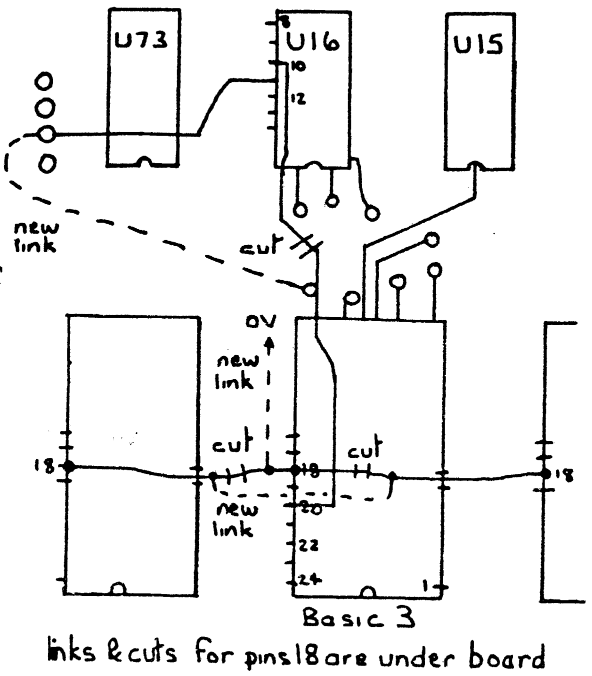
Jeff Rae

GARBAGE COLLECTOR BUG FIX

I have replaced my BASIC 3 chip with a 2716 EPROM which contains a new garbage collector routine by Earl Morris in PEEK(65) June 81.

The article said 'This code appears to be a total solution'.
It has not failed on my 8K Superboard yet.
The following 3 tests all work individually or collectively.

```
 10  REM TEST 1
 20  FORX=1TO25:A$(5)=A$(5)+'A':PRINTLEN(A$(5));:NEXT
 30  :
 40  REM TEST 2
 50  W$(1)='TEST':W$(2)='ING':W$(3)=W$(1)+W$(2)
 60  PRINT FRE(8):PRINT W$(3)
 70  :
 80  REM TEST 3
 85  DIM A$(11)
 90  FORX=65TO90:A$(0)=A$(0)+CHR$(X)
100  PRINT A$(0):PRINT FRE(8):NEXT
```



To replace BASIC 3 with a 2716 on a Superboard 2, three tracks have to be cut and three new wire links added.

1) The signal to pin 20 has to be inverted by feeding it from U16/11 instead of from U16/10.
2) Pin 18 has to be connected to 0V instead of +5V (pin 18 connected to pin 12)
3) The track between BASIC 2 pin 18 and BASIC 4 pin 18 must be reconnected together.

On the original Superboard, U18 can be used to invert the signal to pin 20.
I can supply BASIC 3 EPROMS, programmed as above for $10.00.

John Whitehead

****

MODEMS

If you have been waiting to buy a modem bacause you could not afford it or felt that at three hundred dollars they were too expensive then wait no longer. Dick Smith Electronics have developed a 300 baud full duplex direct connect modem called the DATAPHONE. It is fully Telecom approved and is the first and only one in Australia to have such approval.

The price I will be selling these modems is an amazing $165 to KAOS members. Get your orders in quickly and don't miss out on this exciting stage in communications technology. If you have an order or a question I can be contacted after hours on 470 5027 or write to: Kelvin Eldridge, PO. Box 173, Reservoir, 3073.

By Kelvin Eldridge.

****

FOUND ...... The undocumented directive.
.END. = Terminate Assembly. A common directive in assemblers, but not documented in OSI's version. (Try it, it works.)

David Dodds

MICRO-TRADER

One of the facilities lacking in our hobby industry is a paper like the TRADING POST, but dedicated to the needs of us hobbyists. Needs such as the following:
i/ Selling secondhand hardware and software.
ii/ A place we can advertise hardware and software that we develop and wish to sell ourselves.
iii/ Services that we can offer to others.
iv/ Wanted to buy or exchange.

To this end in August, I will be publishing the first issue of such a paper. The paper I envisage will initially be only a small number of pages growing as word spreads about its existence.

The paper will be sold initially for 4¢/A4 page. I ask you to support this paper as I feel it will be a great benefit to all us hobbyists. The rates for advertising and the classifications are given below.

i/ PAY IF YOU SELL: This section includes any secondhand hardware/software, products that users have developed and retailers advertisements.
10% of the price less than $100 plus 5% on the remainder

ii/ PREPAID ADVERTISEMENTS: This section includes wanted to buy/exchange or any services you wish to offer. eg. EPROM burning, printing listings or programming.
10¢ per word with a minimum charge of $3.00.

If you have any questions you can call me on 470 5027 any night or on the weekend. If you wish to post in your advertisement the address is: MICRO-TRADER, PO Box 173, Reservoir, 3073.

NOTE: All advertisements are the responsibility of the advertiser. I retain the right to withhold publishing any advertisement.

Kelvin Eldridge

****

FOR SALE

SUPERBOARD II, Series 2, 8K of RAM, Power supply and Software. 12 Months old. Excellent condition. $299 Ring and ask for Cameron.

TELETYPE Model 15 in good condition, complete with power supply, interface, local/remote switch, operations handbook and software (Superboard)
$70.00 C. Haustorfer

DISKETTES - VERBATIM: Single-sided double density 525-01 - $3.60 ea.
Double-sided double density 550-01 - $4.55 ea. 8" - from $2.70 ea.
Contact Greg de Vere on
For large quantity, please phone for price.

SUPERBOARD II, DABUG III, 2MHz mod PLUS TASKER BUS with 3 X 8K RAM boards and VIA/PIA board and power supply. Lots of documentation and software.

SPECIAL PRICE - $500.00 Contact KAOS for this bargain.

****

OOOPS!

Rosemary was careless in newsletter No.9 and typed Peter Careless on the Faster Cassette article instead of Peter Carless. Sorry Peter!